EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

2. Authorization for this examiner's amendment was given in a telephone interview with Mr. Gero G. McClellan, Reg. No. 44227, on 07/23/2008.

3. Amend the claims as follow:

a. Replace claim 1 with the following claim:

Claim 1. A computer-implemented method for parallel processing a sequence of requests received by an application server configured to execute a plurality of threads, comprising:

initiating, by the application server, a first thread configured to (i) perform the sequence of requests received from a user interacting with a web-based application and (ii) to maintain state information specific to the first thread, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread configured to perform the sequence of requests and to maintain state information specific to the second thread;

performing, sequentially, by the first thread, the sequence of requests;

performing, sequentially, by the second thread, the same sequence of requests, wherein the second thread performs requests, of the sequence, previously performed by the first thread a specified number of steps behind first thread and while the first thread continues to execute requests, from the sequence of requests, whereby the first thread and second thread each independently maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon completion of the sequence of requests by the first thread, producing a primary result for the sequence of requests;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests;

displaying the primary result produced by the first thread; and

discarding, without displaying, the secondary result produced by the second thread.

b. Cancel claim 5.

c. Replace claim 8 with the following claim:

Claim 8. A computer-implemented method for parallel processing a sequence of requests received by an application server configured to execute a plurality of threads, comprising:

initiating, by the application server, a first thread configured to (i) perform the sequence of requests received from a user interacting with a web-based application and (ii) to maintain state information specific to the first thread, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread configured to perform the sequence of requests and to maintain state information specific to the second thread;

performing, sequentially, by the first thread, the sequence of requests;

performing, sequentially, by the second thread, the same sequence of requests, wherein the second thread performs requests, of the sequence, previously performed by the first thread a specified number of steps behind first thread and while the first thread continues to execute requests, from the sequence of requests, whereby the first thread and second thread each independently maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively; and

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests:

terminating the first thread;

recovering from the error by executing remaining requests of the sequence by the second thread up to, but not including, the request resulting in the detected error;

prompting the user to modify the request which resulted in the detected error;

receiving a modification to the request;

executing the modified request;

executing any remaining requests, of the sequence of requests; and

displaying a final result produced by the second thread from executing (i) the sequence of requests up to, but not including, the request resulting in the detected error, (ii) the modified request and (iii) the remaining requests of the sequence, if any.

d. Replace claim 11 with the following claim:

Claim 11. A computer-implemented method for parallel processing of requests received by an application server configured to execute a plurality of threads, comprising:

receiving a sequence of requests from a user, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

placing the sequence of requests on a queue managed by the application server in a time-ordered manner;

performing, by a first thread managed by the application server, each request, of the sequence, upon being placed on the queue;

performing, by a second thread managed by the application server, at least a portion of the user requests on the queue step-wise with the first thread and N-requests behind the first thread; wherein the first thread and second thread each independently maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon completion of the sequence of requests by the first thread, producing a primary result for the sequence of requests;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests;

displaying the primary result produced by the first thread; and

discarding, without displaying, the secondary result produced by the second thread.

e. Replace claim 21 with the following claim:

Claim 21. A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing a sequence of requests received by an application server configured to execute a plurality of threads, the operation comprising:

initiating, by the application server, a first thread configured to (i) perform the sequence of requests received from a user interacting with a web-based application and (ii) to maintain state information specific to the first thread,

wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread configured to:

perform requests, from the sequence of requests, previously performed by the first thread a specified number of steps behind first thread and while the first thread continues to execute requests, of the sequence; and

maintain state information specific to the second thread, whereby the first thread and the second thread each maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon completion of the sequence of requests by the first thread, producing a primary result for the sequence of requests;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests;

displaying the primary result produced by the first thread; and

discarding, without displaying, the secondary result produced by the second thread.

f. Replace claim 23 with the following claim:

Claim 23. The computer readable storage medium of claim 21, the operation further comprising:

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests, terminating the first thread; and

performing, by the second thread, one or more requests of the sequence previously performed by the terminated first thread and not yet performed by the secondary executing entity.

g. Cancel claim 25.

h. Replace claim 28 with the following claim:

Claim 28. A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing a sequence of requests received by an application server configured to execute a plurality of threads, the operation comprising:

initiating, by the application server, a first thread configured to (i) perform the sequence of requests received from a user interacting with a web-based application and (ii) to maintain state information specific to the first thread, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread configured to:

perform requests, from the sequence of requests, previously performed by the first thread a specified number of steps behind first

thread and while the first thread continues to execute requests, of the sequence; and

maintain state information specific to the second thread, whereby the first thread and the second thread each maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests;

terminating the first thread;

performing, by the second thread, remaining requests of the sequence up to, but not including, the request resulting in the detected error;

prompting the user to modify the request which resulted in the detected error;

receiving a modification to the request;

executing the modified request;

executing any remaining requests, of the sequence of requests; and

displaying a final result produced by the second thread from executing (i) the sequence of requests up to, but not including, the request resulting in the detected error, (ii) the modified request and (iii) the remaining requests of the sequence, if any.

i. Replace claim 29 with the following claim:

Claim 29. A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing requests received by an application server configured to execute a plurality of threads, the operation comprising:

receiving a sequence of requests from a user, wherein each request of the sequence, is composed as a uniform resource locater submitted to the web-based application;

placing the sequence of requests on a queue managed by the application server in a time-ordered manner;

performing, by a first thread managed by the application server, each request, of the sequence, upon being placed on the queue;

performing, by a second thread managed by the application server, at least a portion of the user requests on the queue step-wise with the and N-requests behind the first thread; wherein the first thread and second thread each independently maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon completion of the sequence of requests by the first thread, producing a primary result for the sequence of requests;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests;

displaying the primary result produced by the first thread; and

discarding, without displaying, the secondary result produced by the second thread.

4. The following is an examiner's statement of reasons for allowance:

The cited prior arts of record taken alone or in combination does not fairly suggest or teach the claimed invention of system and method having a first thread and a second thread initiating by the application server to perform the sequence of requests received from a user interacting with a web page application, wherein each of the requests is composed as a uniform resource locator submitted to the web application, wherein the first and second thread configured to maintain its own state information regarding the results of each request of the sequence performed by itself, performing sequentially by the first thread the sequence of requests, performing sequentially by the second thread the same sequence of requests previously performed by the first thread a specified number of step behind the first thread from the sequence of requests while the first thread continues to execute requests , as recited in claims 1, 8, 11, 21, 28 and 29. In addition, prior art of record failed to teach producing a primary result for the first thread upon completion of the sequence of requests, producing a secondary result for the second thread upon completion of the sequence of requests, displaying the primary result produced by the first thread and discarding without displaying the secondary result produced by the second thread as recited in claims 1, 11, 21 and 29. In claims 8 and 28, prior art of record failed to further teach the steps of terminating the first thread in response to an error, recovering from the error by executing remaining requests of

the sequence by the second thread up to, but not including, the request resulting in the detected error, prompting the user to modify the request which resulted in the detected error, receiving a modification to the request, executing the modified request, executing any remaining requests, of the sequence of requests, and displaying a final result produced by the second thread from executing the sequence of requests up to, but not including, the request resulting in the detected error, the modified request and the remaining requests of the sequence. Thus the claims are allowable over the prior arts of record.

5. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jennifer N. To whose telephone number is (571) 272-7212. The examiner can normally be reached on M-T 6AM- 3:30 PM, F 6AM- 2:30 PM.

7. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

8. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Meng-Ai An/
Supervisory Patent Examiner, Art Unit 2195

Jennifer N To
Examiner
Art Unit 2195